TECH-CHRONICLE
AN INTERNATIONAL E-JOURNAL ON
EMERGING TRENDS IN SCIENCE, TECHNOLOGY AND MANAGEMENT

Tracing the Nearest Location By Extracting Appropriate Skyline Points From Partial Data

Snehal M. Khati
Department of Computer Engg.
SSGMCE
Shegaon, India
khatisnehal@gmail.com

Dr. Sameer S. Prabhune
HOD of Information Technology
SSGMCE
Shegaon, India
ssprabhune@gmail.com

Abstract: The skyline query has proven to be an important tool in multi-criteria decision making and search space pruning. Skyline query returns the subset of points from a multi-dimensional dataset that are not dominated by any other point. Due to its wide applications, skyline query and its variants have been extensively studied in the past. However, skyline computation for incomplete domain, where points have missing values for some dimensions, has not received enough attention. The past solutions for such incomplete dataset use weak pareto dominance relation which is non-transitive and cyclic. Hence many of the desirable points are not included in the skyline. Consequently, the skyline no longer offers a reliable overview of the dataset. Moreover, the skyline set returned by these methods is unordered and has high cardinality. The end user does not have control over the result size. Therefore, we have adapted the top-k frequent skyline approach proposed for complete datasets to find interesting points from incomplete datasets. The existing approach overcomes the above mentioned drawbacks and returns top-k points ordered by their fractional skyline frequency. Our proposed approach increase the efficiency of IDFS algorithm and it gives you the nearest location, of shops which fulfills your product requirement, as per your current location.

Keywords: Skyline query processing, partial data, missing data.

I. INTRODUCTION

Owing to its wide applications in multi-criteria decision making and search space pruning, skyline queries have been extensively studied in the last decade. Given a set of points in d-dimensional space S. A skyline query returns all the points that are not dominated by any other point in the set. Here, a point p is said to dominate another point q, if p is better than or equal to q in all dimensions and p is strictly better than q in some dimensions s_i of S. These dominance relation is also known as pareto dominance. Such non-dominated points returned by a skyline query are called skyline points and the set of skyline points is known as the skyline.

Consider a customer looking for a mobile phone based on features such as price, battery backup and screen size. Now, take two phones p_1= ($100, 4hrs, 5in) and p_2= ($200, 4hrs, 4in). Here, p_1 dominates p_2 because each feature of p_1 is at least as good as the corresponding feature of p_2. Such manual examination of all phones is not feasible for large databases. Phones which are not worse than any other phone with respect to all features, are worth considering for the user. Firing a skyline query for such scenario will eliminate all undesirable phones and would present a manageable set of attractive phones to the customer.

Skyline queries and its variants have been extensively studied in the literature. However, with

the exception all skyline query processing approaches assume completeness of data i.e. values corresponding to all dimensions of data points are known beforehand. But, this assumption may not hold in many real world applications such as sensor data, questionnaires and product/service rating websites. In the mobile phone example, the price of certain phone may not be available due to some reasons, making the data points corresponding to that phone incomplete. A modified dominance relation, which is called weak pareto dominance in this paper, has been used in [2,5] to compute skyline for such partial datasets. In weak pareto dominance, while comparing a pair of points, only the subset of dimensions where values are known for both the points, are considered (ignoring dimensions where value is missing for at least one point). Due to the non transitive and cyclic nature of this dominance relation, many of the desirable points are not returned in the skyline.

For example, consider a customer who wants to rent a flat in a city and is browsing a real estate website for the same. Each flat has several features such as area, rent, number of bedrooms, location, distance from the office. It would be difficult and time consuming for the customer to make a choice, if all the flats in the database which are up for rent, are presented to her. Flats which are not worse than any other flat with respect to all features, are of interest to the customer. Firing the skyline query for such scenario will eliminate unwanted flats and would present a manageable set of interesting flats to the user.

Movie rating site (e.g. MovieLens1) is another example where computing skylines could be useful. MovieLens captures, among other things, movie ratings that are given by the users. Now, consider a person interested in highly-rated movies. In such a case, if each user is treated as a dimension then a movie will be a point in the d-dimensional space, where d is the number of users. Computing the skyline will prune all low-rated movies in the data set and return the top-rated movies as the answer.

Consider points m_1, m_2 and m_3 from the sample movie rating datasets shown in Table I. Each entry r_{ij} in the table corresponds to the rating given to movie i by user j on a scale of 1-5. Now, based on weak pareto dominance, m_1 dominates m_2, m_2 dominates m_3, but m_1 does not dominate m_3. Instead, m_3 dominates m_1. Therefore, we can see that, the dominance relation may become non transitive as well as cyclic. Here, none of the three points belong to the skyline, since each one is dominated by at least one other point. Observe that, m_1 is a desirable point (due to reasonably high ratings given by users) even though it is not a skyline point.

One of the shortcomings of traditional skyline query is the large size of the skyline when the datasets has many dimensions. The reason is, for a pair of points p and q, if p is preferred over q in some dimension s_i and q is preferred over p in dimension s_j ($\neq s_i$), then p and q become incomparable i.e, they do not dominate each other. This is the minimum requirement for any pair of points to become incomparable. While computing skyline for high dimensional datasets, we are more likely to find such dimension s_i and s_j. Thus, a large number of points become incomparable with others and belong to the skyline.

Point	u1	u2	u3	u4
m1	2	3	5	-
m2	-	2	5	2
m3	3	-	-	1
m4	1	2	5	2
m5	-	-	-	6

Table I: Sample Movie- rating Dataset

In the mobile phone example, in addition to above mentioned features, consider that memory, operating system and camera resolution are also important for the customer. In such a case, most of the phones may have to be included in the skyline since there may not exist a single phone which is better than other in all the features. However, phones which are better than others in majority of the features may exist but are indistinguishable in the skyline set. Thus, having such a large skyline does not offer any interesting insights into the dataset.

The skyline for partial datasets also suffers from the curse of dimensionality. Weak pareto dominance relation needs just one common dimension with known values to compare a pair of partial data points. Consequently, points having poor values in common dimensions, will not be included in the skyline. Thus, partial data points having few bad values are penalized heavily in weak pareto dominance. Therefore, extracting

manageable set of relevant skyline points from partial datasets is difficult.

One possible way to address this problem is to rank skyline points using some preference function [1]. However, this approach is not always applicable since users need to provide a scoring function i.e., weight assignments pertaining to their preferences. Assigning weights for large number of preferences is difficult and requires sufficient domain knowledge which cannot be expected from an average user. While various approaches that return interesting points without using any scoring function have been proposed for complete datasets [3,6,7,8], the problem of finding and ranking interesting points from partial datasets has not been addressed yet.

II. PRELIMINARIES

In this section, we formally define various concepts related to our algorithm. We would be using sample movie-rating dataset given in Table I as our running example.

Consider an partial dataset D, defined on space S, with dimensions $S=\{s_1, s_2, \ldots, s_d\}$. A point $p \in D$ can be represented as $p=\{p.s_1, p.s_2, \ldots, p.s_d\}$ where $p.s_i$ denotes the value of point p on dimension s_i. Missing value for any dimension of a point is represented as '-'. There are at most $2^d - 1$ distinct non-empty subsets of S. Hereafter, each of them is referred to as a subspace.

A data point $p \in D$ is said to be complete on subspace S' defined by dimensions S' if p has known values for all dimensions in S'.

A. Subspace Skyline

Consider a pair of points p and q that are complete on subspace $S' \subseteq S$ defined by dimensions $S' \subseteq S$. p is said to dominate q if and only if $\forall s_i \in S'$, $p.s_i \geq q.s_i$ and $\exists s_j \in S'$, $p.s_j > q.s_j$.

In other words, we use weak pareto dominance to compare a pair of points having known values for all dimensions in subspace S'. Points that are not complete on S' are not considered for computing skyline of S'. The reason for not using weak pareto dominance relation here, is stated towards the end of this section. Therefore, the skyline of subspace S' is defined as a subset of points in D that are (1) complete on S' and (2) not dominated by any other point on S'.

In Table I, while computing skyline of subspaces S_{12} formed by dimension sets $\{u_1,u_2\}$, we consider only points m_1 and m_4, since they are complete on S_{12}. Here, m_4 is dominated by m_1 and thus, the skyline of S_{12} is $\{m_1\}$.

B. Skyline Frequency

Skyline frequency [4] is one of the interestingness measures proposed for complete datasets. The skyline frequency of a point p, f(p), is the number of unique subspaces $S' \subseteq S$ in which p is a skyline point.

For example, by definition *A*, point m_2 from Table I is a skyline point on subspaces formed by dimensions $\{u_3\}$, $\{u_2,u_4\}$, $\{u_3,u_4\}$ and $\{u_2,u_3,u_4\}$. Thus, $f(m_2) = 4$. Similarly, $f(m_5) = 1$.

Due to the definition of subspace skyline, data points with a large number of missing values will have small skyline frequency. Moreover, skyline frequency of points will not be consistent across the dataset since data points often have varying number of missing values. Therefore, we propose fractional skyline frequency, denoted by f'(p), as the skyline frequency of a point divided by the total number of complete subspaces for p. That is, $f'(p) = f(p) / (2^k - 1)$, where k is the number of known dimensions for point p. The usefulness of this metric follows from the fact that, a point with k (<= d) known dimensions can be dominated in maximum of $2^k - 1$ subspaces. Thus, $f'(m_2) = 4 / (2^3 - 1) = 4 / 7$ and $f'(m_5) = 1 / (2^1 - 1) = 1$.

Intuitively, k points with highest fractional skyline frequency in a dataset are called as top-k frequent skyline points.

C. Dominating Subspace

Given a data point $p \in D$, if $\exists q \in D$ that dominates p on a subspace S' subset of S, then S' is said to be a dominating subspace for m_1 since point m_3 dominates m_1 on S_1.

In Table I, subspace S_1 defined by dimension $\{u_1\}$ is a dominating subspace for m_1 since point m_3 dominates m_1 on S_1.

D. Dominating Frequency

The dominating frequency of a data point p Є D, d(p), is the number of unique dominating subspaces for p.

Intuitively, skyline frequency and dominating frequency are duals of each other. Hence, dominating frequency of the point p can be described as : $d(p) = 2^d - 1 - f(p)$. Analogous to the concept of fractional skyline frequency, we propose fractional dominating frequency as d'(p). Therefore, $d'(p) = 1 - f'(p)$ and we have, $d'(m_2) = 3 / 7$ and $d'(m_5) = 0$.

Thus, computing k points with smallest fractional dominating frequencies in a dataset would be same as computing top-k frequent skyline points. We will be using fractional skyline / dominating frequency for computing top-k frequent skyline points.

E. Dominating Subspace Set

The set of all subspaces on which a point q dominates another point p is known as dominating dominating subspaces of q over p and is denoted by DS (q, p).

Consider points m_1 and m_2 from Table I. m_1 dominates m_2 in subspaces $S_2 = \{u_2\}$ and $S_{23} = \{u_2, u_3\}$. Thus, DS $(m_1, m_2) = \{S_2, S_{23}\}$ and can also be represented as $(U, V) = (\{u_2\}, \{u_3\})$.

III. RELATED WORKS

In this section, we discuss the existing work on traditional skyline queries, its variants and skyline queries for partial domains.

A. Traditional Skyline Queries

Borzsonyi et al. [11] first introduced a Skyline operator for relational databases and proposed BNL, D and C and an algorithm using B tree for skyline evaluation. Since then, various skyline query processing algorithms, using different approaches, have been proposed by the research community. For instance, index structures are used in [6] to progressively report the skyline.

Skyline queries have been proposed for domains such as partially ordered [12], spatial [5], uncertain [9] and distributed databases [2]. Comprehensive survey of skyline query processing in highly distributed environments and uncertain domains are presented in [10, 11] respectively. Unfortunately, all of the above mention skyline evaluation approaches were proposed for complete data and cannot be easily adapted for partial data.

B. Variants of Skyline Queries

The traditional dominance relation is quiet strict, resulting in large size of skyline. In case of correlated datasets, a point good in one dimension tends to be good in all other dimensions as well. Consequently, few "good" points dominate large number of points, resulting in small size of the skyline. Hence, several variants of traditional skyline query have been proposed to control the skyline size and to find the interesting subset of the skyline. These variants employ some interestingness metric to extract relevant points from the dataset. The skyline return by these approaches depend on the dominance relation and interestingness metric use. Some of the most prominent variants are discussed below:

Thick skyline query allows user to increase the skyline size by including points within Є-distance of skyline points. Papadias et al. [6] proposed top-k skyline that ranks skyline points based on user define monotone scoring function. The authors also proposed k- sky band query that returns points that are dominated by at most k other points. This approach allows user to explore other non- skyline points in case the traditional skyline is too small. Zhang et al. [8] proposed the concept of ∂-subspace to find strong skyline points. A subspace whose skyline contains less than ∂ points, is called ∂-subspace and the union of all skyline points in such ∂ subspace is called strong skyline points. The k-dominant skyline query [3] relaxes the dominance relation from considering all dimensions to any subset of size k. Therefore, more points get dominated and a few high quality points can be return. The top-k representative skyline points (top-k RPS) extract the k-points such that the total number of unique data points dominated by them is maximized. Є-skyline enables user to increase / decrease the skyline size by allowing points to dominate other if their normalized values are within a constant of Є. This approach allows user to assign weights and rank points using built-in order. Distance based representation skyline returns the k most representative skyline points of the dataset i.e., the k points that minimizes the maximum distance between a non-representative skyline point and its closest representative.

Unfortunately, all the above variants where proposed in the context of complete data. The dominance relation in all these variants will not work if values corresponding to some dimensions of data points are missing. Hence, any of these approaches, if adapted for partial domain would need weak pareto dominance relation for comparing points. This relation is not preferred since it may discard many desirable points from the result set, explain in section I.

C. Skyline Queries for Partial Data

Khalefa et al. [5] introduced the problem of skyline computation for partial datasets. They defined a modified dominance relation (weak pareto dominance) for comparing partial data points and proposed three algorithms namely Replacement, Bucket, and ISkyline for skyline computation. Among these, ISkyline is incremental in nature and the most efficient. Zhang et al. [8] proposed to convert an partial dataset to corresponding complete dataset by plugging in estimated values for the missing dimensional values. Now, any of the existing skyline computation approaches can be applied to find the skyline over the complete dataset. However, such substitution of values is only viable when incompleteness percentage is small. This approach is also not suitable for critical applications where false result are not acceptable. The Sort-based Incomplete Data Skyline (SIDS) [2] algorithm out performs ISkyline, but does not support incremental addition of data points.

Since both ISkyline and SIDS algorithms use non-transitive and cyclic weak pareto dominance relation, some of the interesting points may not be included in the skyline. Author tried to address this problem by proposing k-sky band queries for partial data. However, some desirable points may still not be included in the skyline, since a point i.e., dominated by k other points on a subspace can have better values on other subspaces. Moreover, this approach returns an unordered set of skyline points and the end user does not have control over the size of the skyline. Afterword, they introduce the another efficient IDFS algorithm i.e., Incomplete Data Frequent Skyline.

IV. PROPOSED APPROACH

In existing approach the IDFS algorithm is used to find the top-k superior skyline points from partial datasets. The naïve approach of computing top-k frequent skyline points for partial datasets is to first compute skyline for each of the 2^d - 1 subspaces using any traditional skyline computation algorithm and then determine the skyline frequency of each point by counting the number of subspaces for which the point belongs to the skyline. Though several efficient approaches have been proposed to compute precise skyline for all subspaces, such a computation is still very expensive.

Chan et al. [2] proposed the frequent skyline metric and an efficient top-k frequent skyline algorithm, based on the concept of MDSS. Their approach first finds the set of MDSSs for a point and then computes the dominating frequency for that point either precisely or approximately. However, this approach was originally proposed for complete datasets. We have adapted it for partial datasets to rank skyline points by fractional skyline frequency and called it as Incomplete Data top-k Frequent Skyline (IDFS).

A. Incomplete Data frequent Skyline

This approach is based on the fact that each dominating subspace of a point p is covered by at least one MDSS for p. Therefore, dominating frequency of p, d(p), is the number of unique subspaces covered by the set of MDSSs ($\mathcal{M}$) for p. Hence, after computing $\mathcal{M}$ and each point, k point with lowest fractional dominating frequency are returned as the top-k frequent skyline points.

The order of processing data points affects the pruning capacity of a skyline algorithm. Thus, Chomicki et al. [13] proposed to sort data points by some monotone sorting function. One such function is to sort data points in non-increasing order of some of their dimension values. The intuition here is that, points with higher sum are likely to have higher skyline frequency and thereby help in early pruning of other points. Therefore, we give pre-sorted data as input to the IDFS algorithm.

Algorithm 1 : IDFS(D, S, k)

Input : d- dimensional partial dataset D, k
Output : ResultSet, the set of top-k frequent skyline points

1. Initialize fractional frequency threshold $\Theta = 1$
2. ResultSet = ϕ

3. foreach p Є D do
4. initialize $\mathcal{M} = \phi$
5. flag = ComputeSetOfMDSS(p, k, Θ, |ResultSet|, $\mathcal{M}$)
6. if flag then
7. d(p) = CountDS($\mathcal{M}$)
8. d'(p) = d(p) / $(2^{\text{dimCount(p)}} - 1)$
9. if (|ResultSet| < k) or (d'(p) < Θ) then
10. if |ResultSet| = k then
11. remove the point with highest fractional dominating frequency in ResultSet
12. end
13. insert p to ResultSet
14. update Θ to be the highest fractional dominating frequency in ResultSet
15. end
16. end
17. end
18. return ResultSet

The pseudo-code of our approach is given in Algorithm 1. The algorithm takes as input, a d-dimensional partial dataset D and the number of frequent skyline pointsin return, k. To avoid explicitly sorting points by their fractional dominating frequency, we initialize Θ with threshold of 1 (Step 1). Θ keeps track of highest fractional dominating frequency in ResultSet. ResultSet is the set of top-k frequent skyline points and is initialized in Step 2.

At the beginning of each iteration, $\mathcal{M}$, the set of MDSSs for a point p, is reset to Θ (Step 4). The algorithm computes the set $\mathcal{M}$ for point p in Step 5 by calling procedure ComputeSetOfMDSS() (given in Algorithm 2). This procedure returns false if p can not be a top-k frequent skyline point and true otherwise. In case p is a potential top-k point, $\mathcal{M}$ would be the set of MDSSs. For such cases, the dominating frequency, d(p), is recomputed by calling CountDS($\mathcal{M}$) (Step 7). The fractional dominating frequency of p, d'(p) is smaller than Θ, then the point with the highest fractional dominating frequency is removed (Step 10-12) and p is initiate into ResultSet (Step 13). The value of Θ is updated in Step 14 to the highest fractional dominating frequency in ResultSet. Finally, the algorithm terminates when all points have been presented and returns ResultSet as the set of top-k frequent skyline points.

B. Computing the set of MDSSs

The procedure ComputeSetOfMDSS() that computes $\mathcal{M}$ for a point p is given in Algorithm 2. The procedure takes k,
Θ and cardinality of ResultSet as input. It returns false if p is determined not to be a top-k frequent skyline point and true otherwise. The procedure compares p with all other points q Є D one at a time and computes the dominating subspaces of q over p, DS(q, p), as a subspace pair (U, V) described earlier. The pair (U, V) is determined by comparing points p and q across individual dimensions (Step 3-4). Here, we note that, only the subset of dimensions where values are known for both the points are considered. Steps 5-15 list two optimizations. Step 17-25 check whether subspace (U, V) is a MDSS or not. By lemma 1, if (U, V)is covered by some MDSS (P,Q) Є $\mathcal{M}$, then (U, V) is not a MDSS for p and is ignored immediately (Step 18-21). Similarly, if (P, Q) is covered by (U, V) then it is not a MDSS and is thus removed from $\mathcal{M}$ (Step 22-24). Finally, if (U,V) is not covered by any (P, Q) Є $\mathcal{M}$ then it is added to set $\mathcal{M}$ (Step 26-28). The procedure terminates once p has been compared with all other points unless terminated early because of optimization.

Algorithm 2: ComputeSetOfMDSS

Input : p, k, Θ, r, $\mathcal{M}$
Output : boolean value and $\mathcal{M}$ *for points p is computed in part or full*

1. initialize chkPoint = 2
2. foreach q Є D / {p} do
3. let U $\subseteq$ S such that $\forall_{s_i}$ Є U, q.s_i > p. s_i
4. let V $\subseteq$ S such that $\forall_{s_i}$ Є U, q.s_i = p. s_i
5. if (r==k) and ((($(2^{|U|} - 1)2^{|V|} / (2^{\text{dimCount(p)}} - 1)$) > Θ) then
6. return false
7. end
8. if (r = k) and (|$\mathcal{M}$| = chkPoint) then
9. d(p) = CountDS ($\mathcal{M}$)
10. d'(p) = d(p) / $(2^{\text{dimCount(p)}} - 1)$
11. if (d'(p) > Θ) then
12. return false
13. end
14. chkPoint = 2 * chkPoint
15. end
16. initialize isMaximal = true
17. foreach MDSS (P, Q) Є $\mathcal{M}$ do
18. if (U U V $\subseteq$ P U Q) and (U $\subseteq$ P) then
19. isMaximal = false
20. break out of the loop
21. end
22. else if (P U Q $\subseteq$ U U V) and (P $\subseteq$ U) then
23. remove (P, Q) from $\mathcal{M}$
24. end
25. end
26. if isMaximal then
27. insert (U, V) into $\mathcal{M}$
28. end
29. end
30. return true

In proposed approach, the efficiency of IDFS algorithm is increased and it gives you the nearest

location, of shops which fulfills your product requirement, as per your current location. Then, as per your location, it selects the nearest shops, which gives you the set of maching products as a result set. This result set shows you the distance of that shop from your current location with direction also.

REFERENCES

[1] R. Bharuka and P. S. Kumar. Finding Superior Skyline Points from Incomplete Data. The 19th International Conference on Management of Data (COMAD), 19th-21st Dec, 2013 at Ahmadabad, India.

[2] R. Bharuka and P. S. Kumar. Finding skylines for incomplete data. In Proceedings of the 24th Australasian Database Conference, pages 109-117, Adelaide, Australia, 2013. Australian Computer Society.

[3] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 503-514, New York, NY, USA, 2006. ACM.

[4] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. On high dimensional skylines. In Proceeding of the 10th international conference on Advances in Database Technology, pages 478-495, Berlin, Heidelberg, 2006. Springer-Verlag.

[5] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski. Skyline query processing for incomplete data. In Proceeding of the 24th international conference on Data Engineering, pages 556-565, Washington, DC, USA, 2008. IEEE Computer Society.

[6] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. ACM Transactions on Database Systems, 30(1):41-82, 2005.

[7] J. Yang, G. P. Fung, W. Lu., X. Zhou, H. Chen, and X. Du. Finding superior skyline points for multidimensional recommendation applications. World Wide Web, 15(1):33-60, 2012.

[8] Z. Zhang, X. Guo, H. Lu, A. K. H. Tung, and N. Wang. Discovering strong skyline points in high dimensional spaces. In Proceedings of the 14th international conference on Information and Knowledge Management, pages 247-248, New York, NY, USA, 2005. ACM.

[9] J. Pei, W. Jin, M. Ester, and Y. Tao. Catching the best views of skyline: a semantic approach based on decision subspaces. In Proceedings of the 31st International Conference on Very Large Data Bases, pages 253-264. VLDB Endowment, 2005.

[10] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. Efficient computation of the skyline cube. Very Large Data Bases, pages 241-252. VLDB Endowment, 2005.

[11] S. Borzsonye, D. Kossmann, and K. Stocker. The skyline Operator. In Proceedings of the 17th International Conference on Data Engineering, pages 421-430, Washington, DC, USA, 2001. IEEE Computer Society.

[12] C. Y. Chan, P. K. Eng, and K. L. Tam. Stratified Computation of skylines with partially ordered. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 203-214, New York, NY,USA, 2005. ACM.

[13] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 297-306, New York, NY, USA, 2000. ACM.